23 October 2000

WBS Dictionary for the US-CMS "Core Applications Software" Sub-Project (WBS items 2.1–2.4)

Draft Version 1.3

Compiled on behalf of

The US-CMS Collaboration

by

Ian Fisk, UC San Diego and

Lucas Taylor, Northeastern University

Summary

This document describes the WBS task breakdown for the "Core Applications Software" sub-project of the US-CMS Software and Computing Project.

WBS 2.1
Software Architecture

It is the intention of CMS to deploy a coherent architecture for all aspects of physics data processing, including simulation, higher-level triggering, reconstruction and selection, physics analysis and visualisation. This includes use of the software both at single computing locations and over wide-area networks. The software architecture project seeks to facilitate this by ensuring that the problems we are solving are well researched and by ensuring that the core on which the software will be built on is well engineered and understood by the community. This involves both evolving the existing core frameworks towards a good, modular set of co-operating components as well as producing a documentation set ranging from use cases to constraints to architectural blueprints to deployment plans.

WBS 2.1.1 CAFE

The CAFE project covers the documentation side of the architecture. It provides tools for the document project, analyses the problem domain, manages the documentation, documents the different aspects of the architecture and feeds input back to architecture to evolve it.

WBS 2.1.1.1 Architectural Document Publishing Tools

This task covers the infrastructure required to produce the entire architectural documentation for a particular project as a web and in printable forms. It involves tools to produce regular releases of entire document set such that it is internally consistent, comprehensively cross-referenced and indexed, and where each document part can be traced to its source and related documents (for instance which particular software packages satisfy a specific requirement, and where that requirement came from in the first place).

WBS 2.1.1.2 Domain Analysis

This task covers research into the problem domain: developing an understanding of the problems we are to solve. It covers activities such as collecting use cases and requirements, understanding constraints, understanding the working models of the physics groups and individuals, and analysing the impact of the possible solutions on the working models (and vice versa).

WBS 2.1.1.3 Document Project Management

Planning, management, editing and releasing of the document sets produced by CAFE.

WBS 2.1.1.4 Architectural Views Documentation

Documenting each major component of the architecture as well as how it will function and be deployed in the distributed environment. Each topic will be covered in the range from a high-level view of a few paragraphs or illustrations to more detailed design diagrams and explanations.

WBS 2.1.1.5 Design Evolution

Propagating the feedback and new ideas back to the architecture implementation.

WBS 2.1.2 Overall Architecture and Framework

The overall (or "core") architecture provides tools and a framework on which different applications such as ORCA will be built on. This task covers the evolution of that architecture, in particular on how the various subsystems will interface to each other through the core framework, CARF.

For the subsystems to be able to talk to each other through the core framework, interfaces are required for items such as persistency (including how subsystems can talk about persistent objects including navigation and browsing, giving hints on expected behaviour, setting user preferences such as input sources and output destinations, dealing with meta-data etc.), the execution model (including how to configure and use the action-on-demand mechanism), the distributed processing (including how to instruct the system about matters such as locating replicas, controlling and configuring policies, deciding on where jobs will run and on what data sets, providing feedback to the user and querying their choices), and visualisation (including how graphical applications can plug into the system and interact with it). Many of these items have more elaborate task breakdowns discussed elsewhere in this document.

WBS 2.1.3 Subsystem Architectures

WBS 2.1.3.1 Detector Geometry Description Sub-Architecture

This task will provide an environment for creating, manipulating, and using the parameters describing the CMS detector in a consistent manner. In particular, it covers the geometrical description of the detector elements at various levels (full engineering detail, full GEANT detail, fast simulation, trigger tower geometries, etc.), associated material properties, magnetic field map, etc. The sub-system will serve a number of clients including OSCAR, Fast Simulation, ORCA, Calibration, and User Analysis Environment.

WBS 2.1.3.1.1 Generic Detector Description Prototype

The task plans to experiment with a generic interface-and-viewpoint framework. Initially it will investigate solutions based on XML files or data models and mediators for CMS-wide data access. Data sources will represent themselves as XML. The framework will allow retrieval of data from any source (OODB, RDB, HTML files, flat ASCII files) using the same semantics. Writing into data sources using these viewpoints will be supported within certain constraints.

WBS 2.1.3.2 Simulation Sub-Architecture

This task covers architecture specific to simulation: the integration of OSCAR, which is GEANT4 based simulation, into the core framework, CARF; the persistent storage of simulated hits in the the data base for digitisation by ORCA; and the addition of hooks needed for event visualisation, as required.

WBS 2.1.3.3 Reconstruction Sub-Architecture

This task covers architecture specific to reconstruction. As more responsibilities of reconstruction are assigned to the Physics Reconstruction and Selection, PRS, groups it may be necessary to assign software professionals to develop fast and efficient software architectures for physicists to insert physics reconstruction algorithms. The on-line reconstruction for triggering and calibration as well as the off-line reconstruction architectures, which are intended to be the same, are difficult working environments requiring a high performance software architecture.

WBS 2.2 Interactive Graphics and User Analysis (IGUANA)

The IGUANA software project addresses interactive visualisation software needs for three domains: graphical user interfaces (GUI's); interactive detector and event visualisation; and interactive data analysis and presentation. The software is to be used in a variety of areas including off-line simulation and reconstruction, data analysis, and test beams. Tasks include the assessment of use-cases and requirements and the evaluation, integration, adaptation, verification, deployment, and support in the CMS environment of visualisation software from HEP, academia, the public domain, and the commercial sector. Given the limited manpower situation, pre-existing software that meets CMS architectural requirements is exploited as much as possible to optimise the use of the resources available.

WBS 2.2.1 Interactive Graphics

This task covers interactive graphical user interfaces and performant 2D and 3D graphics software. It focuses on generic components and is complimentary to the “User Analysis” subtask which includes some CMS-spefici apllications which rely on the “Interactive Graphics” task.

The task includes the provision of an overall architecture for graphical applications, a functional OO GUI toolkit, plus extensions based on it, such as 2D/3D viewers, simple GUIs, tree-like browsers that can be used in more specific applications such as detector and event display, federation database tools, database utilities, test-beams applications, application configurators, and other GUI-based CMS applications that are a part of the “User Analysis”.

This task includes support such as keeping up to date the GUI toolkit installation, providing examples and tutorials to help using it within the CMS collaboration, and research and development in this area.

WBS 2.2.1.1 Graphics Application Architecture

This task covers the general architecture for graphics applications, which must cooperate with the various other (sub-)architectures and frameworks in use by CMS. This task also needs to consider the issues of graphics performance in the context of complex models and distributed processing and data storage resources.

It should provide mechanisms for integrating modules (currently Views), defining module interfaces, and integration with the CMS software framework CARF. One of the current implementations is based on “Model-View-Controller” paradigm (MVC) architecture where Views are separated from Models. There can be many Views within one Workspace. The specific implementation of a Model interface is a task for the “User Analysis”.

WBS 2.2.1.2 Graphical User Interface Tools

This task covers provision of graphical user interface (GUI) tools, in particular a coherent set of widgets spanning the “usual” functionality of modern graphical user interface software.

WBS 2.2.1.2.1 Graphical User Interface Extensions

This task covers extensions to the base GUI toolkits to provide missing or enhanced functionality.

WBS 2.2.1.3 2D Graphics

This task covers the needs for 2D graphics. It includes evaluation of existing low- and high-level 2D libraries which may be part of, or extensions to, GUI toolkits. Tasks include the provision of a set of generic 2D shapes; implementing Cartesian to non-Cartesian transformation; scene graphs; stacking; providing slices and projections; layering; grouping of objects; 3D to 2D conversion etc.

WBS 2.2.1.4 3D Graphics

This task covers the needs for performant 3D graphics. It includes evaluation of existing low- and high-level 3D libraries. Tasks include the provision of a set of generic 3D shapes; scene graphs; stacking; providing slices and projections; grouping of objects, etc.

WBS 2.2.1.5 Viewers and Browsers

This task covers generic viewers of arbitrary and multiple 2D and/or 3D graphics scenes integrated with GUI controllers for controlling program flow and manipulating the scenes interactively.

For example, for 3D scene viewers it includes render areas in which graphics output is drawn; manipulators to control the view parameters, such as magnification, rotation, translation, and slices; tree-like organisers; properties database; extraction of material properties from code to be able to change them without code re-compilation; saving configuration; change background panel; change properties/transparency panels; save/restore viewpoint - small preview panel for selected viewpoint; redirect output to a console window/frame; log file; clone view; drag and drop selected objects from one view to another; interface to request 0-N events, next event, previous event, step over N events; interface to tag selected event; interface to retrieve more information on selected object and print/view it; interface to cancel the request; etc..

WBS 2.2.2 User Analysis

This task covers CMS specific user interface implementation to external projects such as CARF, ORCA, OSCAR, and LHC++. In particular, access to the CMS detector geometry and event data, an architectural design and implementation for integration of the LHC++ components for data analysis, re-use of software that meets the CMS architectural requirements and its integration within the CMS architectural framework.

WBS 2.2.2.1 Detector and Event Display

This is an ongoing task that includes GEANT3 geometry fixes and updates for the CMS detector display; GEANT4 implementation for the above display; OSCAR integration; interaction with CARF framework to load 0-N events; get next event, previous event, step over N event; tag selected event; retrieve more information on selected object; statistics for selected area; cancel the request; reinitialise application - close previous dataset and open another, or close current federated database and open another.

WBS 2.2.2.2 Database Utilities and Federation Tools

This task covers implementation of the access to the objects stored in a federated database and manipulation with them such as move an object, copy an object, delete an object; access to the CMS Meta-data information. This task provides both the base software and GUI implementation.

WBS 2.2.2.2.1 GUI Applications

An example of a simple GUI application is a GUI-based tool to set environment variables and create the initialisation .orcarc file. First implementation will be provided for "Detector and Event Display" (OrcaVis application). When a user starts OrcaVis, he will first face a dialog with a suggestion to specify OO_FD_BOOT if it hasn't been defined. The user will be given an option to browse local disks (file browser dialog). Next, the user will select the Owner and Dataset, filters, etc. When the user acknowledges the changes, the environment variables needed will be set: number of events or interactive, change write/read, etc., the initialisation file .orcarc will be written, and the OrcaVis will be invoked. If the tests and user acceptance will show that it is useful, the GUI will be extended later to enable configuration of other ORCA applications.

Other simple GUIs, for example, a graphical interface that shows a SCRAM project variables grouped in tabs as for "Visual C++ Properties/Options": SCRAM version, list of libraries, include directories, files to be processed by meta-object compiler (MOC), other settings defined for the developers/release area, can be provided for the community.

A GUI for a Database Browser/Viewer: a file system - like access to database objects with a graphical previewer of histograms and tags; drag and drop, delete, copy databases/objects. Different icons for remote/local databases.

WBS 2.2.3 Infrastructure and Support

The main effort has concentrated on creating a flexible infrastructure that can easily handle the import and compilation of external packages (for explanation of why this is needed see above); compilation of user-defined schema and putting them in a federated database as well as creating the database within the project using a project defined scram command; automatic building of the documentation; dependency checking.

WBS 2.2.3.1 QA and Testing

Considering the importance of the CMS requirement for quality software, this task covers the provision of an automatic procedure for analysing the dependencies for external packages, user code, and project software. The software dependencies can be presented as ASCII files or via graphical diagrams.

WBS 2.2.3.2 Project Releases

This task covers installation and testing of a new release for IGUANA and ORCA Visualisation projects. It includes Linux/Sun full build in a test area, running tests, final installation in a release area, automatic creation of the documentation and updating the web pages.

WBS 2.2.3.3 External Software Updates

This task includes installation and testing new releases of Qt Toolkit, import of selected (see above) public-domain software such as new widgets, libraries; providing examples with CMS-specific focus; documentation if it doesn't exist or is insufficient. Evaluation of new LHC++ packages/versions; providing examples how to use them with step by step documentation.

WBS 2.2.3.4 Public-domain Packages

Reuse of software is one of the key issues addressed by the project. For example, the rapidly growing Qt community develops many interesting public-domain applications/libraries/widgets that are generic and which potentially can be useful for the CMS community. This task covers evaluation of new public-domain packages; installation, testing, reuse of parts of code, either importing them in IGUANA or passing to IT for support, or future development.

WBS 2.2.3.5 SCRAM Tool Box

This task covers provision and keeping up-to-date the SCRAM external products descriptions. Currently the descriptions are kept outside SCRAM in the SCRAM Tool Box. This results in a situation where it is possible to add a new version or a new package to the Tool Box without triggering a new release of SCRAM. Given that there will be many different CMS projects using parts of the anther projects code (CMS or not), a consistent CMS tools configuration is needed.

WBS 2.2.3.6 Documentation and Tutorial

This task covers keeping up-to-date the project related documentation; maintaining the link to Qt Tutorial from IGUANA web pages; providing a tutorial for IGUANA preferably Web - based with step by step explanation, examples source code and executable.

WBS 2.2.3.7 Mentoring

This task is based on the CMS requirement for quality software, specifically considering the insufficient level of C++ experience of CMS users. Answering C++ and IGUANA related questions to improve the situation by helping the users to better understand the existing C++ code and enabling them to better write their own C++ code. From the IGUANA team side it includes also the dependency check, helping with redesign, etc.

WBS 2.3
Distributed Data Management and Processing

The Distributed Data Management and Processing software project aims to develop a set of tools, consisting of four pieces, to fully utilise the proposed CMS computing model, which is globally distributed in nature:

- Distributed Process Management
- Distributed Database Management
- Distributed Production Tools
- System Simulation

In the CMS computing model data processing and re-processing, as well as simulations, are distributed over a global grid of computing centres. Some databases need to be replicated and synchronised over all centres and, in situations where it is impractical or impossible to replicate a full database, processes need to be remotely submit to the centre where the data is located.

WBS 2.3.1 Distributed Process Management

A complete system which can efficiently control and schedule jobs over the computing grid is being developed. Eventually it will handle the submission, monitoring, and control of tasks at Tier1, Tier2, and Tier3 regional centres. The task scheduling will be combined with the database replicator to form a system that can send data to jobs or jobs to data depending on which is more efficient.

WBS 2.3.1.1 Distributed Task Scheduling Functional Prototype

Work is in progress on a functional prototype of the Task Scheduling System to allow the submission, termination and monitoring of jobs as a group on a cluster of diverse computers. The system currently has a scheduling mechanism that allows selection of the location to which jobs are submit based on processor type, load, and availability of data set and maintains replicated states, so that computations will not be lost if a server fails. The initial service was successfully tested on 32 processors with the ORCA production software. Attempts to expand the system to 64 processors ran into scalability issues, which have been solved. Within the development of the prototype, and using tools from the MONARC simulation toolkit, advanced job scheduling is being investigated using neural network and traditional techniques

WBS 2.3.1.2 CMS Distributed Task Management Product

Building on what was learned during the functional prototype phase of Task Scheduling, a CMS product will be developed to perform Distributed Task Management. The final product will be integrated with the database replication product to allow the option of moving the data when necessary or more efficient.

WBS 2.3.2 Distributed Database Management

This task described the development of tools external to the ODBMS that control replication and synchronising of databases over the grid as well as performance improvement and monitoring of database access.

WBS 2.3.2.1 Database Replication

Tools are being developed to replicate and synchronise databases between regional centres. As production and analysis become more distributed and less CERN-centric, efficient tools to move databases become more important. This process started with the development of tools to easily replicate databases generated at CERN for use in user analysis at Fermilab. In the functional prototype, databases generated at several sites can be automatically exported so the complete database catalogue stays synchronised over the grid. In the final version, some databases will need to be automatically synchronised and others will need to be exported on demand to maximise the efficiency of distributed analysis.

WBS 2.3.2.1.1 Database Replication Investigative Prototype

The first attempt at automatic database replication tools came in the form of an investigative prototype written in Perl. Calling Objectivity applications, the scripts could make the contents of a Federated Database accessible via the web. The operator could then select the database files to be transferred and they would be sent using scp to the remote site, automatically attached to the Federated Database there, and archived to an MSS system if required. The prototype was written as a versioned product with good documentation and the final development release was from summer 2000. Development work is finished and it is expected to be replaced by more advanced GDMP tools in the Fall of 2000.

WBS 2.3.2.1.2 GDMP Tools Functional Prototype

Grid Data Management Pilot (GDMP) tools are Globus Middleware based database replication tools, which provide all the functionality of the Perl prototype plus an improved interface, automatic web publishing of transfer progress, and the ability to resume a transfer from a checkpoint in case of network failure. In addition GDMP tools will be able to handle automatic replication and synchronisation of databases at registered sites. The first prototype was completed in July of 2000. Version 1.0 was released at the end of summer 2000 and version 1.1 was released at the beginning of October. The final functional prototype is expected in the summer of 2001.

WBS 2.3.2.1.3 CMS Database Replicator Product

Starting from GDMP tools functional prototype, this should become the product that CMS uses to replicate and synchronise databases over the grid to Tier1, Tier2, and Tier3 regional centres.

WBS 2.3.2.2 Performance Tools and Optimisation

The availability and performance, both writing and reading, of the database servers is a serious limitation to the overall performance of analysis and production computing farms. Elements in this section work to improve performance and evaluate bottlenecks. Improving the performance and availability has unfortunately become a project which “puts out fires”. It has been difficult to predict what the limitation of the computing farms will be until they are encountered. Both performance improvement and system monitoring should be turned over to the User Facility sub-project as production becomes less developmental.

WBS 2.3.2.2.1 Objectivity AMS plugins

The performance and capabilities of the Objectivity AMS server can be enhanced by writing ‘plugins’ that conform to a well-defined interface. To improve the availability of database servers, one such plugin called the ‘Request Redirection Protocol’ is being implemented. When the Federated Database determines that an AMS server has crashed (e.g. because of disk failure), jobs can be routed to an alternate server. This should significantly increase the availability of the database servers and subsequently the uptime of the farm. The initial implementation of the Request Redirection Protocol is already in use in the Fall 2000 production. Another AMS plugin, which will implement a security protocol, should be available in the early part of 2001.

WBS 2.3.2.2.2 System Monitoring Tools

To improve the performance of the computing farm, database server studies have been performed on disk performance as a function of number and type of access. For production, studies have been performed on the number of servers necessary to keep the computational nodes busy using the low performance servers that are currently available for CERN production. Tools to monitor and measure performance of the computing farm have been developed. The first use of these tools gave unique input into the MONARC simulations. Production studies using MONARC simulations of production continue, the results should influence the hardware choices made by future production facilities.

WBS 2.3.3 Distributed Production Tools

Tools are developed to facilitate production. At the moment this effort runs in parallel to the Distributed Task Management sub-project and concentrates on developing tools for use in production immediately on existing computer farms. US-CMS production has primarily been performed on computers that were not purchased for CMS and therefore there is large variation in the configuration, platform, and capabilities. It is often necessary to share these production facilities with other projects and there is not the control that one would expect over a CMS dedicated system. Lessons learned in this project are applicable to Distributed Task Management, but this task is expected to be a temporary effort to solve an immediate problem.

WBS 2.3.3.1 Automated Archiving and Transfer

Tools are being developed to automatically record and archive results of production performed remotely and to transfer these results to the CERN Mass Storage System (MSS). This has been used primarily for archiving CMSIM production performed at remote sites. These tools are based on the prototype of WBS 2.3.2.1.1, and are currently used at CERN, Padua, Moscow, IN2P3, Caltech, and Fermilab.

WBS 2.3.3.2 Automated Job Submission and Scheduling

Tools are developed to submit jobs in a generic way over varying production systems. US-CMS is performing remote production on a wide variety of computing systems, from dedicated HP super-computing clusters to clusters of Linux PC's that run processes parasitically. Tools are developed to try to standardise submission to improve the ease of use.

WBS 2.3.4 System Simulation

Simulations of large scale production systems are performed to help understand and optimise performance using the MONARC simulation toolkit. Successful simulations of the Spring 2000 CMS ORCA production have been performed with detailed monitoring of the production for input to the simulation. Simulations will be performed of the 2000 Fall Production trying to build on what was learned. In 2001 there will be a simulation of tape access to evaluate modes and guidelines of tape access and what will be the quality of the service. Also in 2001 there will be a simulation of transactions to update meta-data in the data-base and the effect on production.

WBS 2.3.4.1 MONARC

The MONARC collaboration began in 1998 with the charge of modelling large scale computing systems for use in LHC experiments. The MONARC toolkit has been used for modelling the performance of the computing farm during the Spring 2000 CMS ORCA production with remarkable accuracy. As an indication of the maturity of the simulation tool kit, it is now possible to evaluate task submission schemes on a simulated production system. There are currently on going evaluations of self-organising neural networks for job scheduling in distributed computer systems.

WBS 2.4 Support

One of the primary goals of hiring dedicated software professionals to work within the CMS Software and Computing project was to offer support to US users and developers so that they would be able to participate fully in CMS physics. The Support project attempts to provide easy access to software engineers for help with development; attempts to guarantee that US users are supported with reasonable documentation, examples and training; and attempts to ensure that US developed software is versioned, properly configured, and easily distributed.

WBS 2.4.1 Developer Support

This task seeks to provide developers of detector reconstruction, physics reconstruction, and analysis software help from professional software engineers with implementing, optimising, and porting their code. Below are a few examples of areas that the current group of CAS engineers could be instrumental to off-project developers.

- Detector Geometry and Detector Reconstruction, Michael Case
- Physics Reconstruction, Hans Wenzel
- Analysis Tools, Ianna Gaponenko and Lassi Tuura
- Integration and Framework, Lassi Tuura

WBS 2.4.2 User Support

This section outlines the kind of support that the CAS engineers will provide to US-CMS physics users. This includes access to current documentation, working examples of CAS developed applications, and CMS software training.

WBS 2.4.2.1 Documentation

Support seeks to ensure that documentation for all publicly released CMS software packages are current and available to CMS physicists. This includes the preparation, in electronic form, of documentation for CMS specific software (ORCA, IGUANA, OSCAR, Distributed Data Management and Processing tools, etc.). Generation of documentation should be part of each development project, consolidation and web publishing can be the task of the Support Level 3 project manager.

WBS 2.4.2.2 Examples

This task seeks to provide a set of reliable and current examples for the use of the packages necessary to perform physics analysis in CMS: User Analysis Package (IGUANA) example, a well documented ORCA coding example, eventually a full analysis stream example, etc. Off-project developers should provide examples for their projects. CAS engineers and the Level 3 project manager should verify that the examples are current and useful.

WBS 2.4.2.3 Training

In order for US physicists working in CMS to be successful they must be trained to use and develop software in the CMS environment. This involves the participation of all CAS engineers in yearly CMS Software Training at a US institution for new users on ORCA, IGUANA, CARF, the user environment, and software packages necessary for physics success.

WBS 2.4.3 Software Support Tools

For software to be useful it must be available and distributable. This task will ensure that all US-CMS developed software is versioned and distributed using the CMS agreed upon software support tools: cvs and SCRAM

WBS 2.4.3.1 Configuration Management and Distribution Systems

This task ensures that software packages produced by CAS are consistently versioned and controlled by a distribution management system (cvs) and can be compiled on stand-alone installations using a building system (SCRAM). Creating consistent software versions, putting the code into a software management system, and creating the build files are the responsibility of developers. CAS engineers and the Level 3 project manager should verify that this is being done.